

Vzorový príklad pre MIPSIM
Princípy počítačového inžinierstva 2023
Tomáš Meravý Murárik

Zadanie

Napíšte program, ktorý bude simulovať hru Pac-Man na hracom poli podľa obrázka. Ľavé horné políčko hracieho poľa má súradnice(riadok, stĺpec) = (1,1) a pravé spodné políčko má súradnice (5,5). Na hracom poli sa nachádzajú rozmiestnené bodky. Ak hráčvstúpi na políčko s bodkou, skonsumuje ju a táto bodka zmizne. Za každú skonsumovanú bodku získa hráč 50 bodov. Hráč môžezačínať hru na niektorom z voľných políčok a môže vykonávať kroky o 1 políčko smerom na niektorú svetovú stranu.

	1	2	3	4	5
1	o		o		o
2			H		
3	o		o		o
4					
5	o		o		o

V pamäti údajov (PÚ) uchovávaťe riadkovú súradnicu hráča na adrese **a0h** a stĺpcovú na adrese **b0h**. Od adresy **0h** so 4-bajtovými rozostupmi (4h, 8h, ch, 10h, 14h, 18h, 1ch, 20h, atď.) bude pred spustením programu v pamäti údajov uložená postupnosť hodnôt reprezentujúcich pohyby hráča o 1 políčko nasledovne:

- **1h** – pohyb hore,
- **2h** – pohyb vpravo,
- **3h** – pohyb dole,
- **4h** – pohyb vľavo,
- **0h** – koniec.

Po načítaní hodnoty **0h** sa program ukončí. Môžete predpokladať, že v postupnosti sa iné čísla ako **0h-4h** nebudú nachádzať.

Riešenie

Pamäť programu

Adr.	Label	Inštrukcia	Komentár
0h		LW \$25,00a0(\$0)	načítame začiatočnú riadkovú súradnicu z PÚ z adresy a0h do registra R25
4h		LW \$26,00b0(\$0)	načítame začiatočnú stĺpcovú súradnicu z PÚ z adresy b0h do registra R26
8h	zac	LW \$22,0000(\$20)	do registra R22 načítame prvok postupnosti z PÚ z adresy, na ktorú ukazuje ukazovateľ v registri R20
ch		ADDI \$20,\$20,0004	zvážšime ukazovateľ v registri R20 o 4, aby ukazoval na ďalší prvok postupnosti v poradí
10h		NOP	
18h		BEQ \$22,\$1,jedna	ak je načítaný prvok postupnosti v reg. R22 rovný 1 (konštantu 1 máme uloženú v reg. R1) skoč na podprogram pre vykonanie pohybu hore ktorý sa nachádza na labeli „jedna“
1ch		BEQ \$22,\$2,dva	ak je načítaný prvok postupnosti v reg. R22 rovný 2 (konštantu 2 máme uloženú v reg. R2) skoč na podprogram pre vykonanie pohybu vpravo ktorý sa nachádza na labeli „dva“
20h		BEQ \$22,\$3,tri	ak je načítaný prvok postupnosti v reg. R22 rovný 3 (konštantu 3 máme uloženú v reg. R3) skoč na podprogram pre vykonanie pohybu dole ktorý sa nachádza na labeli „tri“
24h		BEQ \$22,\$4,styri	ak je načítaný prvok postupnosti v reg. R22 rovný 4 (konštantu 4 máme uloženú v reg. R4) skoč na podprogram pre vykonanie pohybu vľavo ktorý sa nachádza na labeli „styri“ inak to musí byť 0 a pokračujeme ďalej:
...	...		
40h	jedna	SUBI \$25,\$25,0001	zmenší y o 1
44h		BEQ \$25,\$25,porovnan	skočí na porovnan
...	...		
54h	dva	ADDI \$26,\$26,0001	pridá 1 k xovej súradnici
58h		BEQ \$0,\$0,porovnan	pôjde na porovnan
...	...		
70h	tri	ADDI \$25,\$25,0001	pridá 1 k yovej súradnici
74h		BEQ \$0,\$0,porovnan	pôjde na porovnan
...	...		
88h	styri	SUBI \$26,\$26,0001	zmenší x o 1
8ch		BEQ \$0,\$0,porovnan	pôjde na porovnan
...	...		
d0h	porovnan	BEQ \$25,\$2,zac	skontroluje či sa riadková alebo stĺpcová súradnica rovná 2 alebo 4, ak áno tak ide na zac
d4h		BEQ \$25,\$4,zac	
d8h		BEQ \$26,\$2,zac	
dch		BEQ \$26,\$4,zac	
e8h		BEQ \$0,\$0,nasobeni	ak sa nenachádzame na súradnici o ktorej vieme že sa na nej nenachádza bod tak ideme na nasobenie
...	...		
124h	nasobeni	NOP	
128h		MUL \$10,\$25,\$5	vynásobíme yovú suradnicu x 10 a pridáme ju na \$10

134h		ADD \$10,\$10,\$26	pripočítame do \$ 10 súradnicu xovú takže \$10 bude vyjadrené iba pomocou jednej premennej
140h		BEQ \$0,\$0,vypoc	skočí na vypocet
...	...		
150h	vypoc	NOP	
		LI \$17,00c0	vložím do R17 hodnotu c0
15ch	calc	NOP	
160h		LW \$16,0000(\$17)	načítam hodnotu z R17
16ch		BEQ \$10,\$16,porov	porovnáam ju z hodnout R10 (kontrolujem či sa nachádzam na bodke)
170h		ADDI \$17,\$17,0004	dám aby pointer 17 ukazoval na ďalšiu hodnotu
174h		ADDI \$13,\$13,0001	pripočítam k môjmo loop counteru 1
180h		BEQ \$13,\$12,reset	porovnáam či je môj loop na konci , ak áno tak idem na reset
184h		BNEQ \$13,\$12,calc	ak nieje loop na konci tak ho dám na začiatok
...	...		
1b8h	reset	LI \$13,0000	vynulujem loop counter
1bch		LI \$17,00c0	dám R17 nech sa znovu pozerá na začiatok
		BEQ \$20,\$20,zac	idem na začiatok
...	...		
208h	porov	NOP	
20ch		ADDI \$15,\$15,0050	pripočítam k môjmu počtu bodiek ďalšiu bodku
210h		SW \$5,0000(\$17)	do memory na R17 vložím číslo na , ktoré sa nemôžem dostať
218h		BEQ \$14,\$15,kon	ak mám v sebe maximálny počet bodov tak idem na koniec
21ch		BNEQ \$14,\$15,reset	ak nie tak idem na reset , ktorý ma zoberie na začiatok
...	...		
140h	kon	SW \$25,000a(\$0)	koniec programu
		SW \$26,00b0(\$0)	

Simulujeme napríklad takúto postupnosť krokov:1 4 4 3 3 3 2 2 2 1 1 1 1 4 3 3 4. kde hráč začína na políčku 2,3 podľa obrázka nižšie:

	1	2	3	4	5
1	↓	←	←	↓	←
2	↓		H	↓	↑
3	↓			←	↑
4	↓				↑
5	→	→	→	→	↑

↑↓←→

Obsah registrov a pamäti údajov pred spustením programu

R16	00000000	address	data memory			
R1	00000001	00000000	00000001	00000004	00000004	00000003
R2	00000002	00000010	00000003	00000003	00000003	00000002
R3	00000003	00000020	00000002	00000002	00000002	00000001
R4	00000004	00000030	00000001	00000001	00000001	00000004
R5	00000010	00000040	00000003	00000003	00000004	00000000
R6	00000000	00000050	00000000	00000000	00000000	00000000
R7	00000000	00000060	00000000	00000000	00000000	00000000
R8	00000000	00000070	00000000	00000000	00000000	00000000
R9	00000000	00000080	00000000	00000000	00000000	00000000
R10	00000000	00000090	00000000	00000000	00000000	00000000
R11	00000000	000000a0	00000002	00000000	00000000	00000000
R12	00000000	000000b0	00000003	00000000	00000000	00000000
R13	00000000	000000c0	00000011	00000013	00000015	00000031
R14	00000009	000000d0	00000033	00000035	00000051	00000053
R15	00000000	000000e0	00000055	00000000	00000000	00000000
R16	00000000	000000f0	00000000	00000000	00000000	00000000
R17	000002d0	00000100	00000000	00000000	00000000	00000000
R18	00000000	00000110	00000000	00000000	00000000	00000000

Register	Údaj	Komentár
R1	1h	konštanta 1 na porovnávanie
R2	2h	konštanta 2 na porovnávanie
R3	3h	konštanta 3 na porovnávanie
R4	4h	konštanta 4 na porovnávanie
R5	10h	konštanta na násobenie
R10	10h	uchovávanie premennej ktorá sa vypočíta pomocou $Y*10 + x$ takže ak sme na súradnici 1 a 3 tak naše R10 sa bude rovnať 13
R13		hovorí o tom na akej pozícii sa náš for loop na kontrolovanie bodiek nachádza
R14	2d0h	konštanta na porovnávanie
R15		počet bodov
R16		hodnoty ktoré sú uložené na adrese R17
R17	c0h	pointer na to , kde sú uložené naše body
R20	0h	ukazovateľ do postupnosti prvkov, na začiatku ukazuje na 1. prvok
R22	0h	sem sa bude načítavať prvok postupnosti z pamäte údajov
R25	0h	sem sa načíta začiatková riadková súradnica z PÚ z adresy a0h následne sa bude počas behu programu aktualizovať
R26	0h	sem sa načíta stĺpcová súradnica z PÚ z adresy b0h následne sa bude počas behu programu aktualizovať

Adresa	Údaj	Komentár
0h – 40h	1h, 1h, 1h, 2h, 2h, ..., 0h	postupnosť krokov
a0h	2h	začiatková riadková súradnica
b0h	3h	začiatková stĺpcová súradnica

Obsah registrov a pamäti údajov po spustení programu

Pc	00000000	R16	00000033	address	data memory				
R1	00000001	R17	000000d0	00000000	00000001	00000004	00030004	00000000	
R2	00000002	R18	00000000	00000010	00000003	00000003	00000003	00000002	
R3	00000003	R19	00000000	00000020	00000002	00000002	00000002	00000001	
R4	00000004	R20	0000004c	00000030	00000001	00000001	00000001	00000004	
R5	00000010	R21	00000000	00000040	00000003	00000003	00000004	00000000	
R6	00000000	R22	00000004	00000050	00000000	00000000	00000000	00000000	
R7	00000000	R23	00000000	00000060	00000000	00000000	00000000	00000000	
R8	00000000	R24	00000000	00000070	00000000	00000000	00000000	00000000	
R9	00000000	R25	00000003	00000080	00000000	00000000	00000000	00000000	
R10	00000033	R26	00000003	00000090	00000000	00000000	00000000	00000000	
R11	00000000	R27	00000000	000000a0	00000002	00000000	00000000	00000000	
R12	00000009	R28	00000000	000000b0	00000003	00000000	00000000	00000000	
R13	00000004	R29	00000000	000000c0	00000010	00000010	00000010	00000010	
R14	000002d0	R30	00000000	000000d0	00000010	00000010	00000010	00000010	
R15	000002d0	R31	00000000	000000e0	00000010	00000000	00000000	00000000	
				000000f0	00000000	00000000	00000000	00000000	
				00000100	00000000	00000000	00000000	00000000	
				nnnnnn1n	nnnnnnnn	nnnnnnnn	nnnnnnnn	nnnnnnnn	

V záverečnej analýze tohto riešenia môžeme konštatovať, že prúdové spracovanie (stream processing) bolo efektívne využité, predovšetkým pri kontrole vstupnej postupnosti krokov a porovnávaní s aktuálnou pozíciou hráča. Využitie podmienených skokov (BEQ) na základe načítaných hodnôt z postupnosti prvkov umožnilo dynamické riadenie programu podľa konkrétnych krokov hráča.